ALGORITMOS Y ESTRUCTURAS DE DATOS
Laboratorio 9

Se busca realizar ejercicios de conceptos asociadas a programación, manejo de estructuras no lineales e interacciones con el sistema de archivos además de la salida y entrada estándar.

Objetivos

- Realizar los ejercicios utilizando el lenguaje de **programación C++.**
- Realizar una correcta creación y manipulación de estructuras.
- Realizar una correcta implementación de algoritmos que involucren decisiones y bucles.
- Realizar una correcta implementación de estructura de datos no lineales.

Nomenclatura para nombre de archivos fuentes

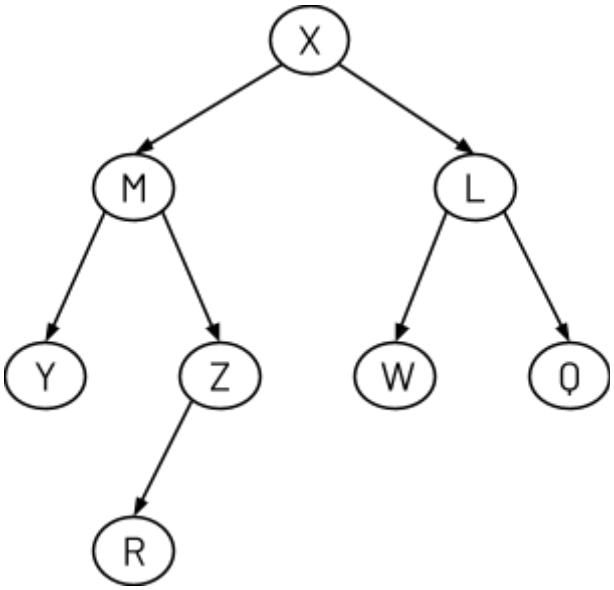
El nombre del archivo en el cual se almacena el código fuente debe considerar el siguiente formato: laboratorioN.EXT donde; EXT es la extensión del lenguaje de programación utilizado y N el número del Laboratorio.

Ayuda memoria

Recursive Traversals / Recorridos recursivos

Crear una estructura que permita trabajar con este tipo de árbol. El recorrido en árboles binarios involucra visitar cada nodo sólo una vez y podemos encontrar tres métodos comúnmente utilizados:

- pre-order: Vista la raíz, recorre el subárbol de la izquierda en pre-order y luego recorre el subárbol de la derecha en pre-order.
- in-order: Recorre el subárbol de la izquierda en in-order, luego la raíz y luego recorre el subárbol de la derecha en in-order.
- post-order: Recorre el subárbol de la izquierda en post-order, recorre el subárbol de la derecha en in-order y luego la raíz.

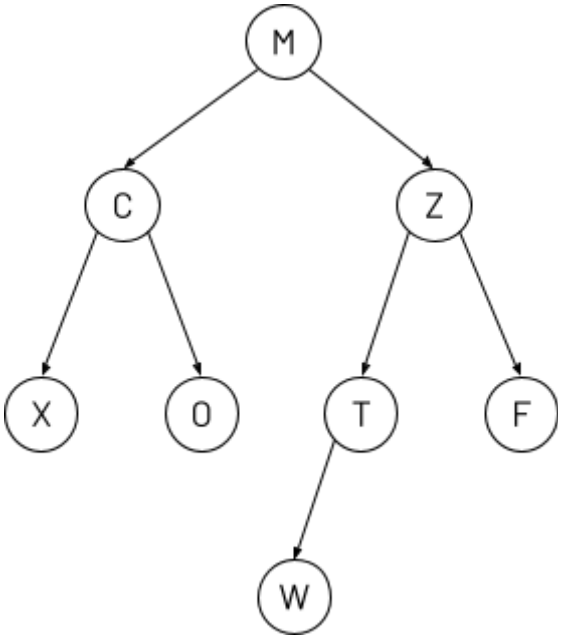


pre-order: X M Y Z R L W Q

in-order: Y M R Z X W L Q

post-order: Y R Z M W Q L X

Ahora tome de referencia el siguiente árbol binario



Ejercicio número 1

Crear una estructura que permita trabajar con este tipo de árbol y crear las instrucciones que permitan incorporar la información al árbol en el orden señalado en la imagen de referencia.

Ejercicio número 2

Crear las funciones para recorrer el árbol en pre-order, in-order y postorden.

Ejercicio número 3

Considere el arreglo entregado en este ejercicio para; implementar la funcionalidad para la creación del árbol binario y la selección del recorrido para que la impresión sea en orden.

Arreglo: 21, 18, 6, 9, 10, 7, 19, 15, 12, 5.

Ejercicio número 4

Crear una función que implemente la búsqueda por profundidad. Aquí se puede considerar la recursividad para realizar la implementación o la utilización de un stack.

Ejercicio número 5

Crear una función que implemente la búsqueda por amplitud. Aquí debe utilizar una cola o queue para realizar la implementación.

Ayuda memoria

```cpp
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* left;
    Node* right;
};

Node* createNode(int data) {
    Node* newNode = new Node;
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}
Node* deleteNode(Node* root, int key) {
    if (root == NULL) {
        return root;
    }

    if (key < root->data) {
        root->left = deleteNode(root->left, key);
    } else if (key > root->data) {
        root->right = deleteNode(root->right, key);
    } else {
        if (root->left == NULL) {
            Node* temp = root->right;
            delete root;
            return temp;
        } else if (root->right == NULL) {
            Node* temp = root->left;
            delete root;
            return temp;
        }

        Node* temp = root->right;
        while (temp->left != NULL) {
            temp = temp->left;
        }

        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }

    return root;
}
```

```cpp
#include <iostream>
#include <queue>
using namespace std;

int main() {
    queue<int> q;
    q.push(21);
    q.push(22);
    q.push(24);
    q.push(25);

    int num=0;
    q.push(num);

    q.pop();
    q.pop();

    while (!q.empty()) {
        cout << q.front() <<" ";
        q.pop();
    }
}
```

```cpp
#include <iostream>
#include <stack>
using namespace std;

int main() {
    stack<int> s;
    s.push(21);
    s.push(22);
    s.push(24);
    s.push(25);

    int num=0;
    s.push(num);

    s.pop();
    s.pop();

    while (!s.empty()) {
        cout << s.top() <<" ";
        s.pop();
    }
}
```